Foundations of Intelligent Learning Agents

PROGRAMMING ASSIGNMENT 3

Autonomous Driving

Name : Balbir Singh

Roll No : 22M0747

Email : 22M0747@iitb.ac.in

0. General Observation Regarding Both Tasks

1. All angles are in degree and measured from x axis in clockwise direction.

2. The given coordinate system in questions is not a natural one i,e (x,y) are positive in fourth Quadrants. To convert into a cartesian coordinate system i.e (x,y) in the first quadrants I simply replaced y with -y .(easy for convenience).

3. It was experimentally observed that the target has its center located at (350, 25) instead of (350,0) according to question. In my modified coordinate system the point becomes (350,-25).

1 .TASK 1 : **The Parking Lot Problem**

The steps to solve the problem are elaborated below

1. Find the orientation angle of Car (The direction in which the car is moving). This can be found easily i.e **theta = state[3].**

2. Find the angle of car with the target location center.
    i. car(x,y) = state[0],state[1]
    ii. target(x,y) = 350,-25 (Actual one observe from the experiment)
    iii. **theta_** can we easily find by these two coordinates

3. Rotate the car until its orientation angle **theta** becomes equal to the angle of car with the target location center **theta_**

4. Accelerate the car till it reaches the target (350,-25) location.

2 .TASK 2 : **The Parking Lot Problem Intensifies**

The Idea here is : Check for the obstacles, If there are any, move to another direction till we find clear path angle between car and target location

The steps to solve the problem are elaborated below

1. In case car is below center (here center means x-axis)
    a. Check for upward path

    b. If there are no obstacles, move the car in an upward direction until it reaches near the center and then rotate the car in the direction of the target and then accelerate till it reaches to end.

    c. If there are obstacles (this obstacle will be above the car), move the car into the right direction until there is no obstacle, when there are no obstacles move the car upward until it reaches the center and then move towards the target.

2. In case car is above center,
    a. Check for downward path

    b. If there are no obstacles, move the car in downward direction until it reaches near the center (here center means x-axis) and then rotate the car in the direction of the target and then accelerate till it reaches to end.

c. If there are obstacles (this obstacle will be below the car), move the car into the right direction until there is no obstacle, when there are no obstacles move the car downward until it reaches the center and then move towards the target.

Implementation Explanation

TASK 1 : The Parking Lot Problem

Rotation Method : Take (state, x_,y_) where the state is [x,y,velocity,angle] and x_ and y_ is coordinate of target. It will return the angle **theta_** value i.e angle between car and coordinate (x_,y_).
Initially it was implemented to return a **step** variable which is the number of steer actions either clockwise or anticlockwise to make car orientation toward the coordinate (x_,y_). But later by observation it is found that angle changes in one steer action are not exactly 3 degrees.

```
def rotation(state,x_,y_):
    x = state[0]
    y = -state[1]
    theta = state[3]
```

```
            theta_rad = math.atan((y_-y)/(x_-x))

            theta_ = (180*theta_rad)/math.pi
            if theta_<=0:
                theta_=abs(theta_)
            else:
                theta_=360-theta_

                    step  =  round(abs((theta  -
theta_)/3))
            return theta_
```

Rotate Method: This function will convert the car orientation toward the angle **theta_.**
In the implementation part, if theta_ is not near to theta it will take steer action appropriately that will lead close to **theta_ .** Otherwise it will simply return the action which will accelerate the car.

```
def rotate(state,theta_):
            theta = state[3]
                  if  ((abs(theta-theta_)<=3)  or
(abs(theta-theta_)>=357)):
```

```
            action = [1,4]
        elif(theta<theta_):
            action = [2,2]
        else:
            action = [0,2]
        return action
```

Get_action method: Take state as an argument and return which action to take next.

```
def get_action(state):
        theta_ =rotation(state,350,-25)
        return rotate(state,theta_)
```

TASK 2: The Parking Lot Problem Intensifies

Coordinate Method: This method will take the center of square obstacles and give the coordinates of the four points of the square obstacles.

```
    def coordinate(x,y):
        ls = []
        y=-y
```

```
            ls.append([x-50,y-50])
            ls.append([x+50,y-50])
            ls.append([x+50,y+50])
            ls.append([x-50,y+50])

            return ls
```

Rotation method: This method is similar as in Task 1

rotate method:

functionality is similar as in Task1 with r modification which will handle the different
Obstacles cases.
self.flag is used to check that if car reache to near the center, if it reaches simply find the current orientation (theta) with target and just accelerate after that.

```
        def rotate(state,theta_):
            theta = state[3]

                    if  ((abs(theta-theta_)<=3)  or
(abs(theta-theta_)>=357)):

                  if (state[1] < 25 and state[1] >
-75):
                      self.flag = 1
```

```
            action = [1,4]
        elif(theta<theta_):
            action = [2,2]
        else:
            action = [0,2]

        return action
```

Check_obstacle: This method will check whether there are obstacles in the path of the car.

```
    def
check_obstacle(state,obs1,obs2,obs3,obs4):
        y = -state[1]
        x = state[0]
        y_mistake =-25
        limit =20

        #first quadrant
        if y>=y_mistake and x>0:
            if x>obs1[0][0]-limit and x<
obs1[1][0]+limit:
```

```
                    #print("test")
                    if y > obs1[2][1]:
                        return True
            # second quadrant
            elif (y>y_mistake and x<0):
                if x>obs2[0][0]-limit and x<
obs2[1][0]+limit:
                    if y > obs2[2][1]:
                        return True
            #third quadrant
            elif(y<y_mistake and x<0):
                if x>obs3[0][0]-limit and x<
obs3[1][0]+limit:
                    if y < obs3[2][1]:
                        return True
            #fourth quadrant
            elif(y<y_mistake  and x>0):
                if x>obs4[0][0]-limit and x<
obs4[1][0]+limit:
                    if y < obs4[2][1]:
                        return True
            return False
```

Get_action : This method will return which action should car take next.

```
        def get_action(state):
            if state[1] < 25 and state[1] > -75:
                theta_ =rotation(state,350,-25)
                return rotate(state,theta_)
            else:
                theta_ = theta_up
            return rotate(state,theta_)
```

Other Variable used:

self.flag

theta_up

```
            initial_state = state
            if initial_state[1] > -25:
                theta_up = 270  #to move upward
            else:
              theta_up = 90   # to move downward
```

Location where the get_action is called

```
if(check_obstacle(state,obs2,obs4,obs3,obs1)):
                    action=rotate(state,0)
                else:
                    if self.flag==1:
                        action = [1,4]
                    else:
                      action = get_action(state)
```